# Cellular Automata Crystal Growth

**Overview -** Simulation of crystal growth is an important aspect in understanding some hidden features of growth mechanisms and its relationship with thermodynamic properties. It is important to understand the basic role of the parameters involved in the growth process and their effects at microscopic level to control the distribution of intrinsic impurities. At microscopic level crystals consist of regular arrays of atoms laid out much like the cells in cellular automation.

**Implementation –**

The crystal growth simulation has been done using object-oriented model in Java

**Classes Involved**

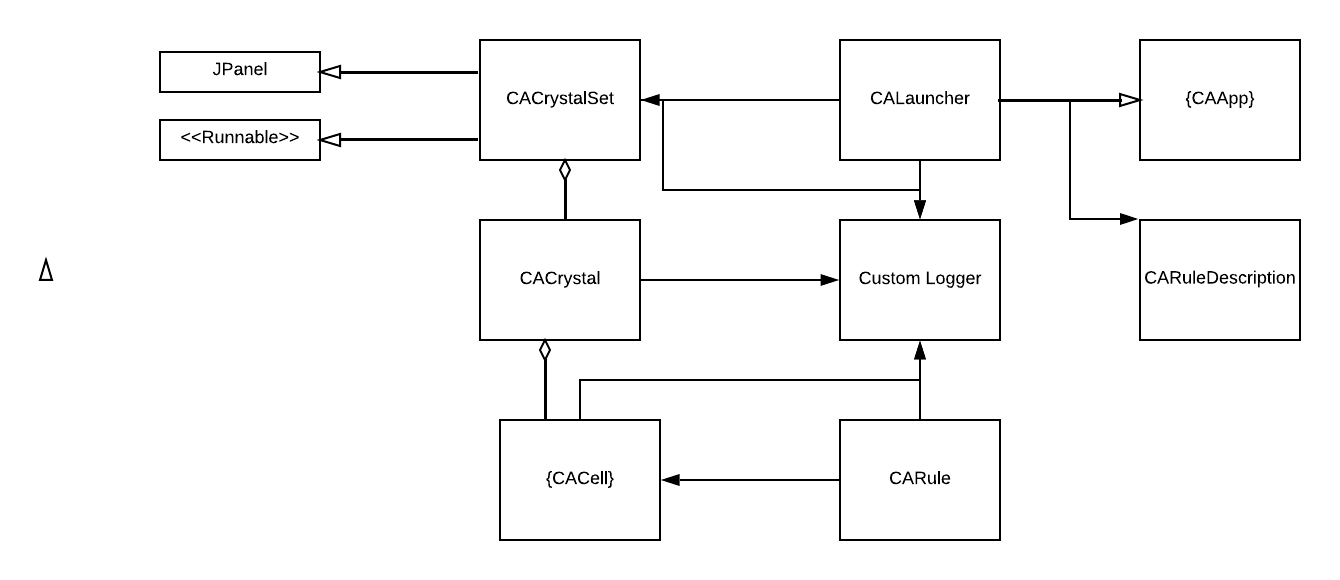
1. **CALauncher –** It is the starting point of the application which extends from CAApp abstract class and implements the abstract method getMainPanel(). It also initializes all the UI components in the sidePanel.
2. **CACrystalSet –** It holds all the crystal generated in every generation in a hash map and it also paints the grid with the crystals. It also implements runnable, so it has thread control functionality as well.
3. **CACrystal -** It contains a 2D array of CACell and rule names. It has the functionality to generate next crystal based on current crystal. It also initializes cell positions for different groups.
4. **CACell –** It is an abstract class and contains cell state of each cell in the form on CACellState enum data type. It has the logic to generate the next cell state based on current cell state and logic to get the neighbour count.This class is derived by the CARule class
5. **CARule –** It extends from CACell and it has logic to get next cell states based on the rule selected. It also has different logics for different rules
6. **CustomLogger –** This class has been created for the purpose of providing single point of creating and instantiating loggers and file handlers for all the classes based on class names
7. **CARuleDescription:** The main purpose of this class is to create frame and display description of each rule when a user clicks on rule description button

**Enumerations Involved:**

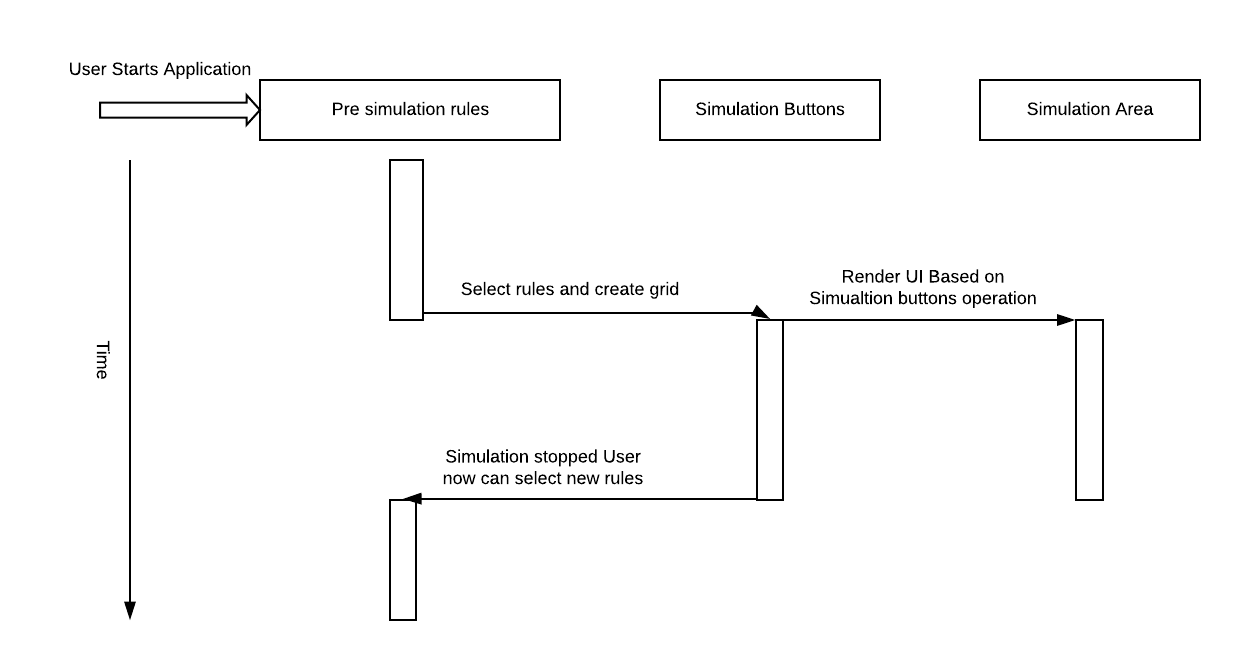
**RuleNames:**  This enum is used represent all the rules used in the application which are {Single Crystal, Random Crystals, Butterfly, rule 4}

**CACellState:** Frozen, Liquid, Vapour are the three states used in the application which signify cell state

**UML Class Diagram**



**Sequence Diagram**



**Program Flow**

CAApp

initGUI()

CACrystalSet

main()

User Interaction Starts

initGUI()

CARule

getNextCellState()

User Clicks on Create after choosing the config (Rules etc.)

*CACell*

*Return next cell sate*

CACrystalSet

Save crystals in Map

New Cell State Returned

User Clicks on Start

CACrystal

createNextCrystal()

run()

**User Interaction**

**Stop**

Simulation Buttons are now disable and back to original state

**Rewind**

Simulation is rewinded from the point where it was paused.

**Pause**

Simulation is paused

Run the JAR file

Select the rule, sleep time, No of generations, Grid/No Grid, Night/dark mode and Click on create

**Start**

Use the simulation buttons to start/pause/stop/rewind

User can restart from the point of pause

After Starting user can

Pause the simulation

# Use Cases

**Use Case 1**: Single Crystal

**ID**: 1

**Level:** Low

**Description:** In this rule, initially only once cell in the center will be Frozen and in the next iteration if only one cell around the cell in the previous crystal is Frozen the state of the cell be transitioned to Frozen in the next iteration

**Actors:** Frozen Cells -> Indicated in blue

**Stakeholders & Interests:** All the cells in the region

**Pre-Conditions:**

1. Rule to be selected by the user is Single Crystal

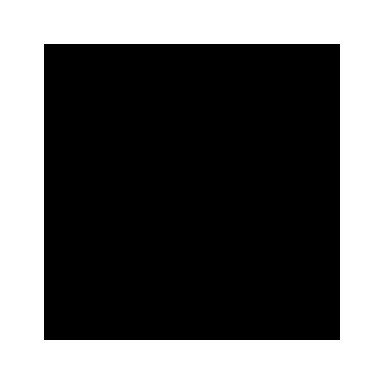
2. Create the initial conditions using Create Button of the side panel.

**Triggers:**

1.  Start Icon -> Start Simulation

2.  Pause Icon -> Pause Simulation

3.  Rewind Icon -> Rewind Simulation

4.  Stop Icon -> Stop Simulation

**Post-Conditions:**

1. **Success**: User is shown “Simulation Completed Successfully” message in status Panel and the console.

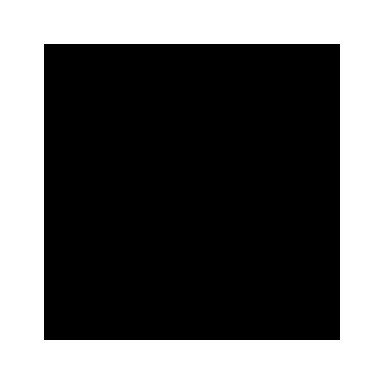
2. **Generation Limit Reached**: When Generation Count reaches Generation Limit, a status “Simulation reached maximum generation Limit…” is shown both in the console and status panel.

3. **Initial State:** When user clicks on  &  and goes back to initial state, Simulation is paused and a status “Simulation Paused as user went back to initial state” is shown. Now User can start or stop the simulation.

**Main Scenario:**

**STEP 1:** Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

**STEP 2:** button is pressed, and simulation is started and finished successfully

**STEP 3:** button is pressed, and simulation is stopped and simulation buttons are disabled and new set of parameters can be selected

**Alternate Scenarios:**

**Scenario 1:**

STEP 1: Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

STEP 2: button is pressed, and Simulation is started.

STEP 3: button is pressed, and Simulation is paused.

STEP 4:  button is pressed, and Simulation goes back from the generation it was paused to previous generation, reaches initial state and simulation is paused again.

**Scenario 2:**

STEP 1: Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

STEP 2: button is pressed, and Simulation is started.

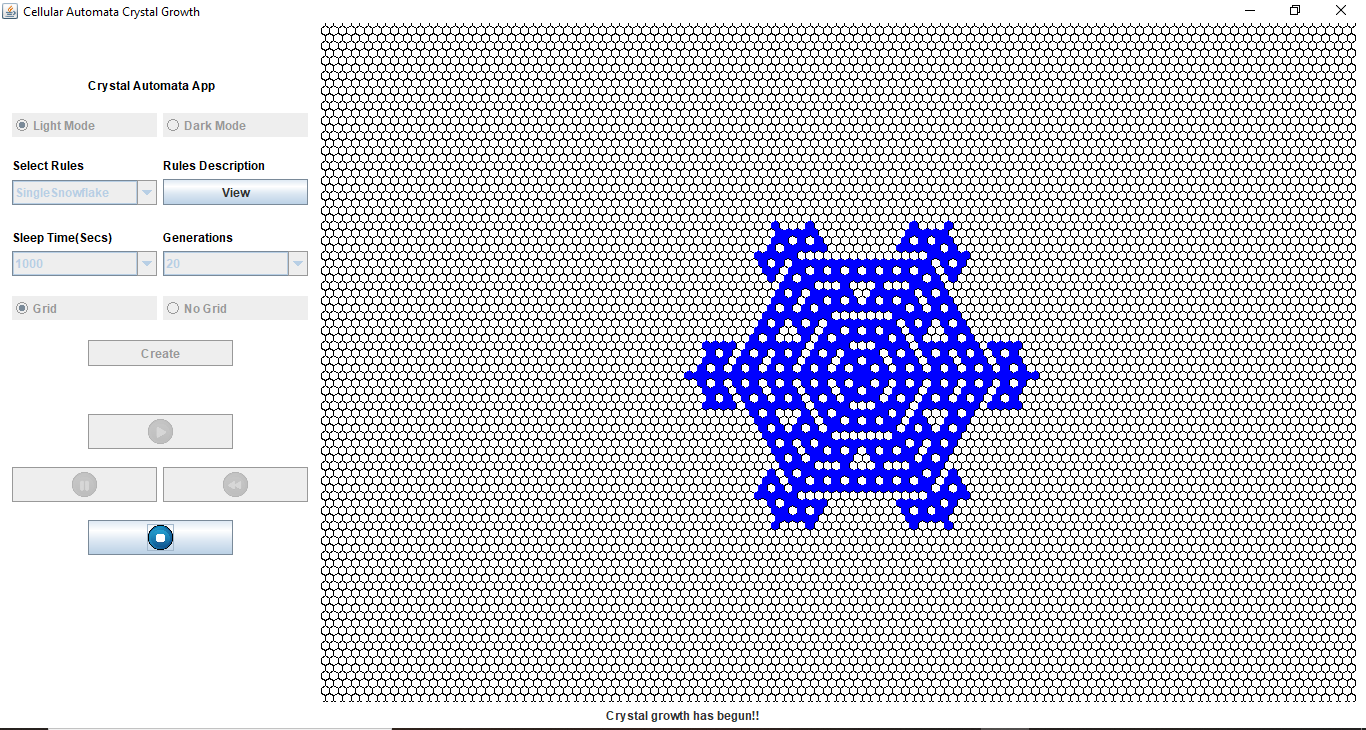
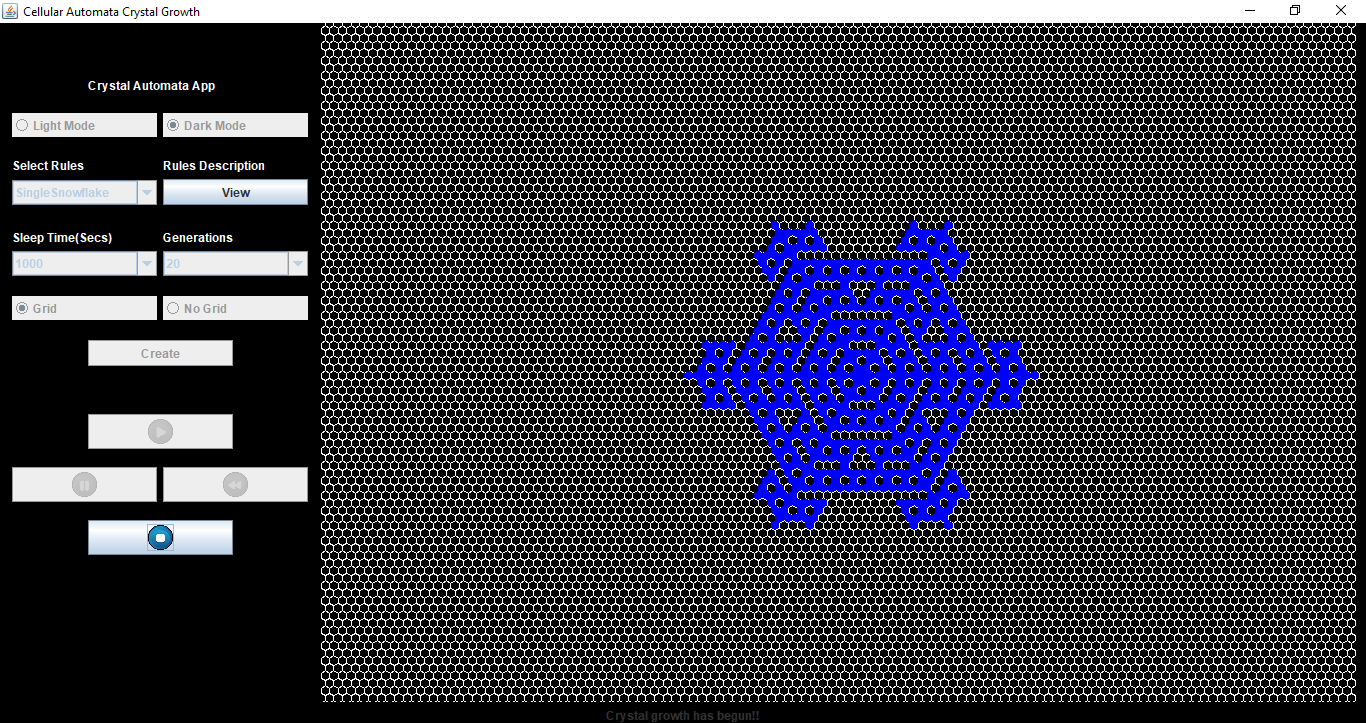
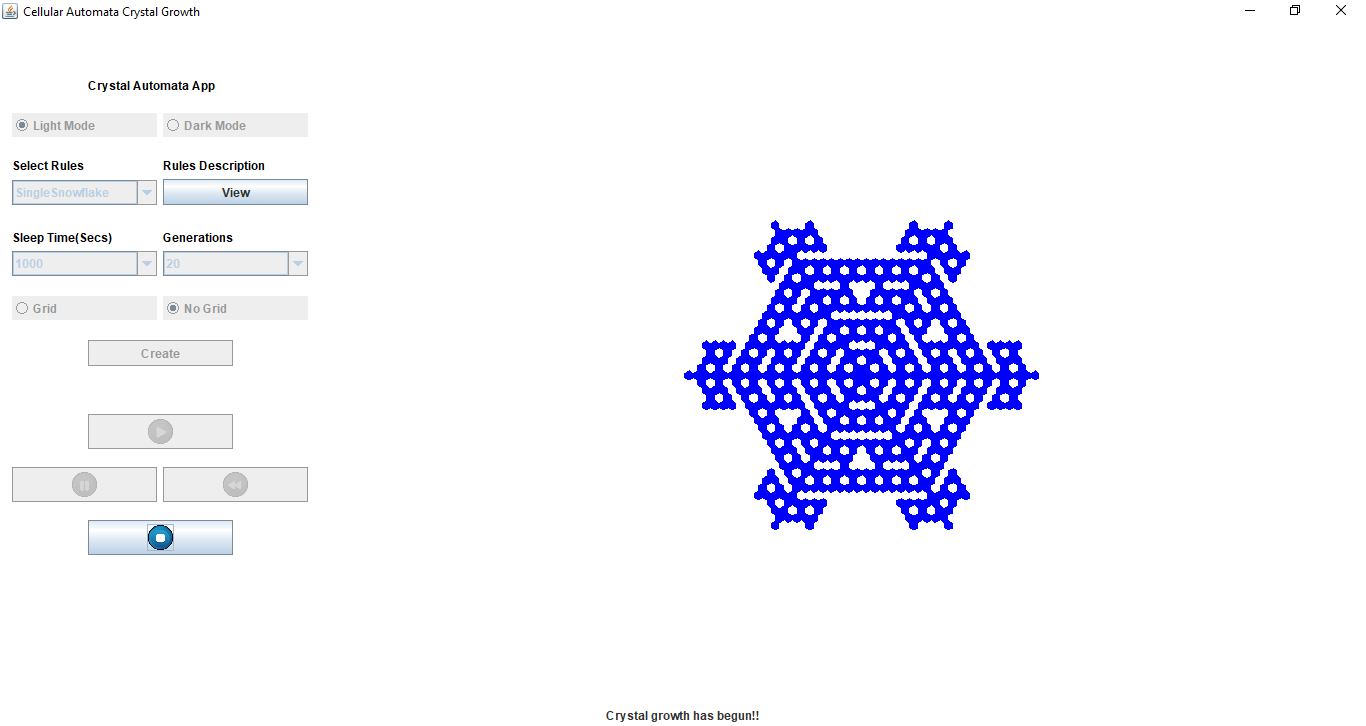
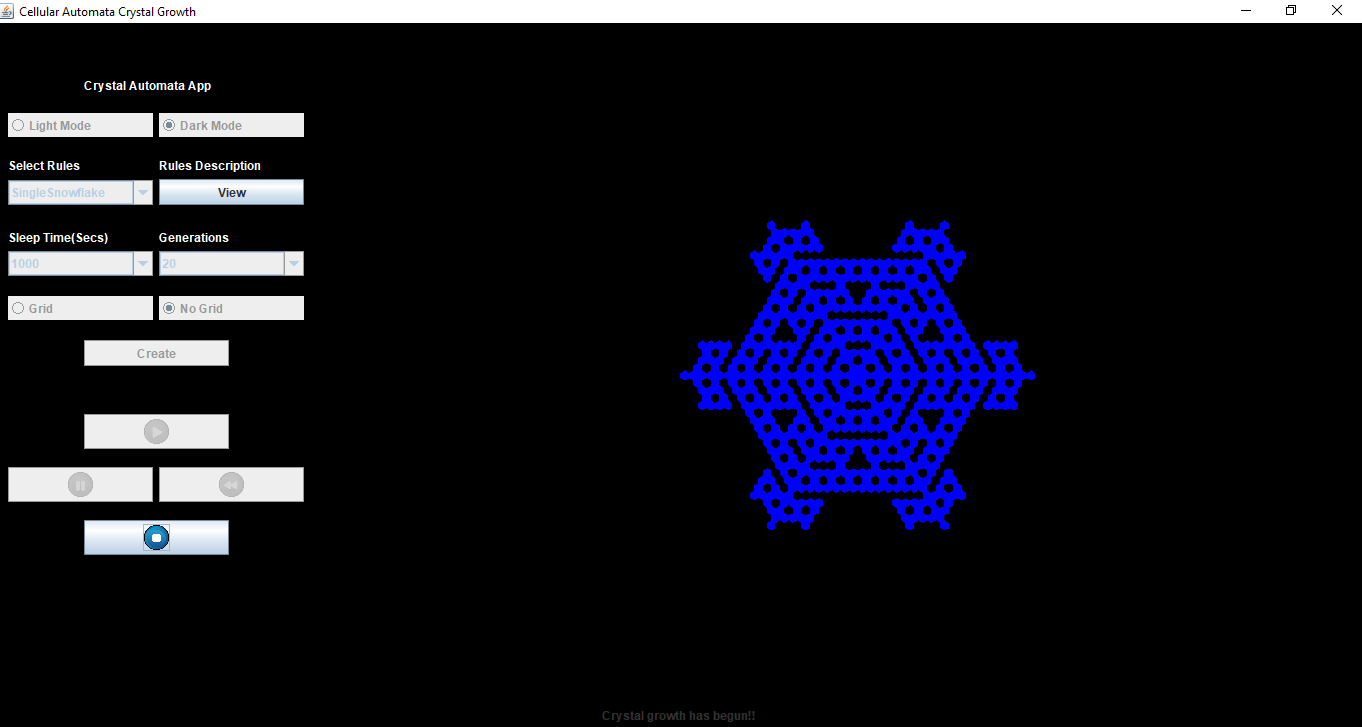
STEP 3: button is pressed, and Simulation is paused.

STEP 4:  button is pressed, and Simulation goes back from the generation it was paused to previous generation.

STEP 5: button is pressed, and Simulation is paused.

STEP 6: button is pressed, and Simulation is started from the point it was paused.

Following are snapshots of the final simulation results with Grid or without grid and in Light/Dark mode



**Use Case 2**: Random Crystals

**ID**: 2

**Level:** Medium

**Description:** Cells in the nature are not always uniform. Through this use case I would like to simulate random crystal like in nature and do the simulation. In this rule, initially 15 random liquid state cells and 10 random Frozen cell states are created in the grid.

Logic used:

**if** (getDesiredNeighborsCount(CACellState.***FROZEN***) == 1 || **this**.getCellState() == CACellState.***FROZEN***) {

**return** CACellState.***FROZEN***;

}

**if** (getDesiredNeighborsCount(CACellState.***FROZEN***) > 3 && **this**.getCellState() == CACellState.***LIQUID***) {

**return** CACellState.***FROZEN***;

}

**if** (getDesiredNeighborsCount(CACellState.***VAPOUR***) > 4 && **this**.getCellState() == CACellState.***LIQUID***) {

**return** CACellState.***LIQUID***;

}

**if** (getDesiredNeighborsCount(CACellState.***VAPOUR***) < 3 && getDesiredNeighborsCount(CACellState.***VAPOUR***) > 1 && **this**.getCellState() == CACellState.***VAPOUR***) {

**return** CACellState.***LIQUID***;

}

**Actors:** Frozen Cells -> Indicated in blue, Liquid state cells -> Green

**Stakeholders & Interests:** All the cells in the region

**Pre-Conditions:**

1. Rule to be selected by the user is Random Crystal Growth

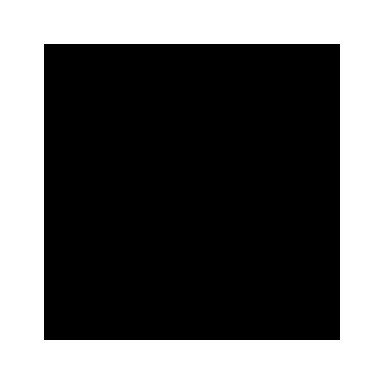
2. Create the initial conditions using Create Button of the side panel.

**Triggers:**

1.  Start Icon -> Start Simulation

2.  Pause Icon -> Pause Simulation

3.  Rewind Icon -> Rewind Simulation

4.  Stop Icon -> Stop Simulation

**Post-Conditions:**

1. **Success**: User is shown “Simulation Completed Successfully” message in status Panel and the console.

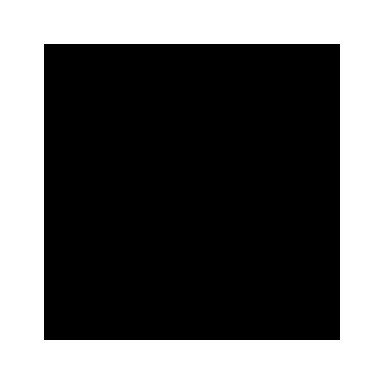
2. **Generation Limit Reached**: When Generation Count reaches Generation Limit, a status “Simulation reached maximum generation Limit…” is shown both in the console and status panel.

3. **Initial State:** When user clicks on  &  and goes back to initial state, Simulation is paused and a status “Simulation Paused as user went back to initial state” is shown. Now User can start or stop the simulation.

**Main Scenario:**

**STEP 1:** Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

**STEP 2:** button is pressed, and simulation is started and finished successfully

**STEP 3:** button is pressed, and simulation is stopped and simulation buttons are disabled and new set of parameters can be selected

**Alternate Scenarios:**

**Scenario 1:**

STEP 1: Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

STEP 2: button is pressed, and Simulation is started.

STEP 3: button is pressed, and Simulation is paused.

STEP 4:  button is pressed, and Simulation goes back from the generation it was paused to previous generation, reaches initial state and simulation is paused again.

**Scenario 2:**

STEP 1: Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

STEP 2: button is pressed, and Simulation is started.

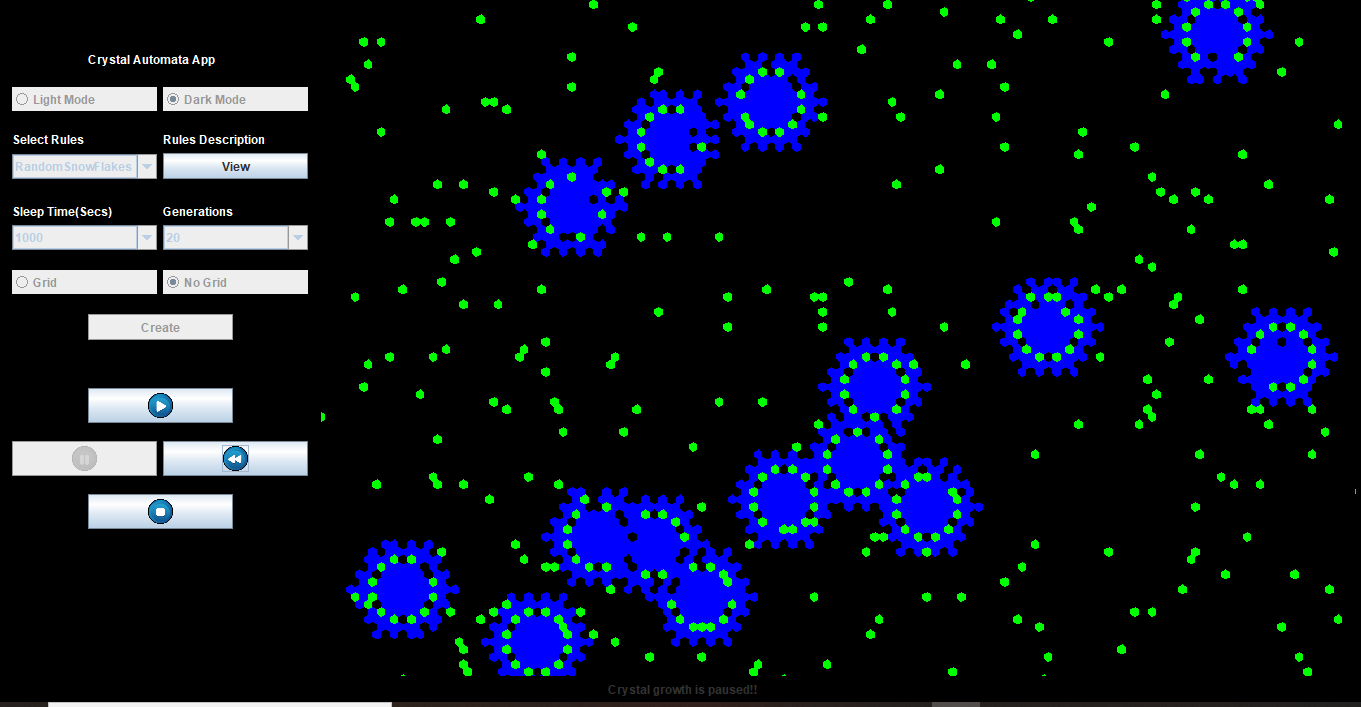
STEP 3: button is pressed, and Simulation is paused.

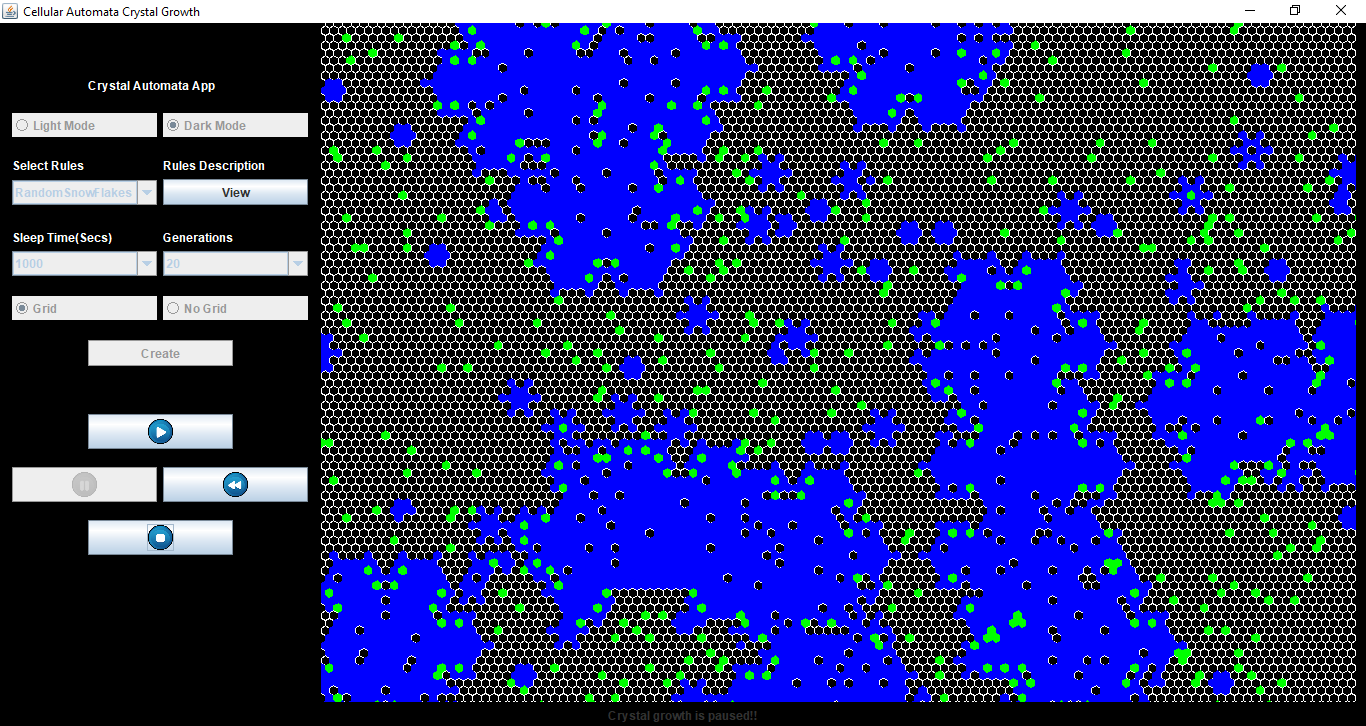
STEP 4:  button is pressed, and Simulation goes back from the generation it was paused to previous generation.

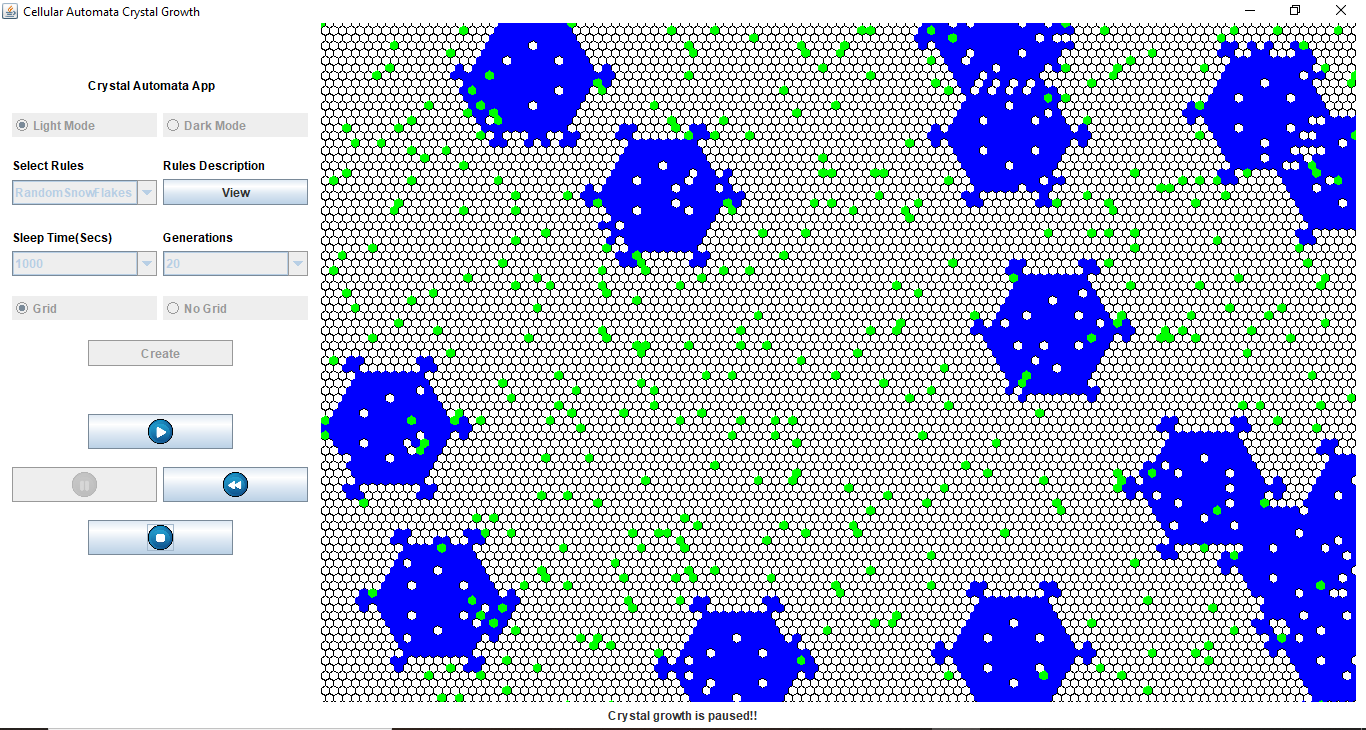
STEP 5: button is pressed, and Simulation is paused.

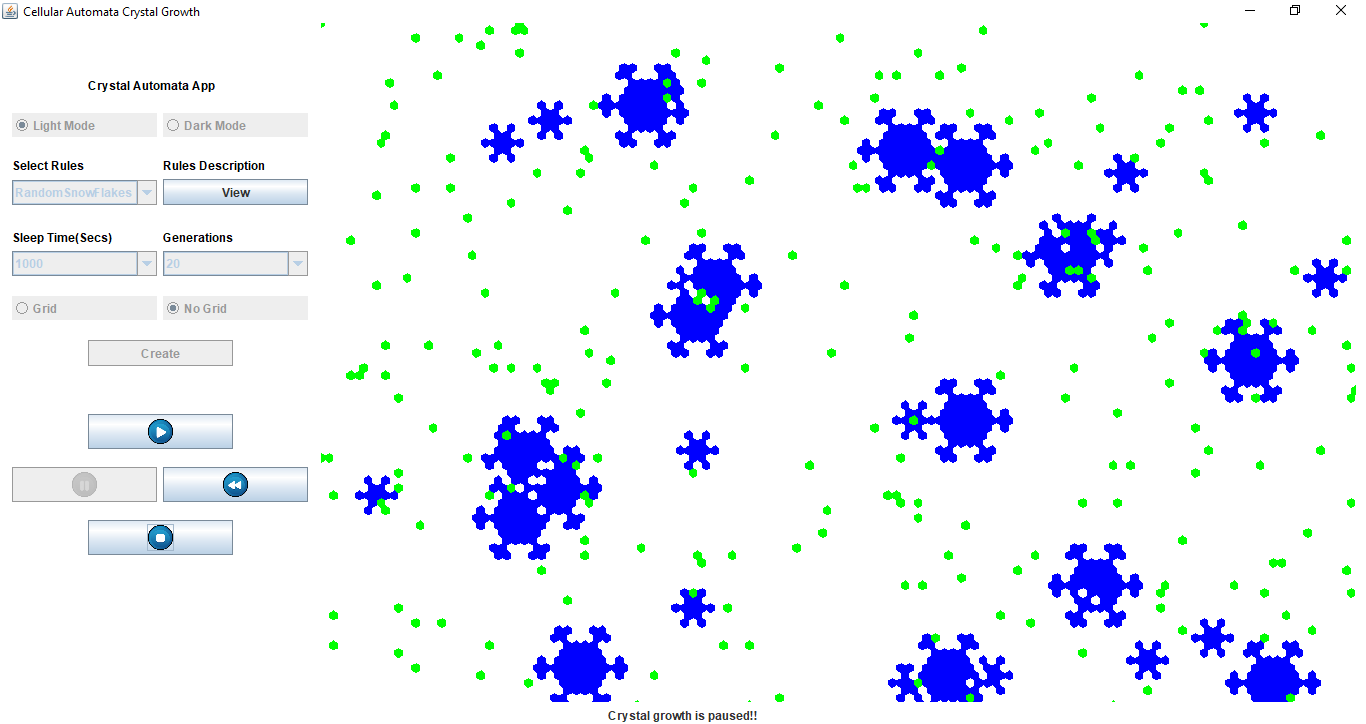
STEP 6: button is pressed, and Simulation is started from the point it was paused.

Following are snapshots of the final simulation results with Grid or without grid and in Light/Dark mode









**Use Case 3**: Butterfly

**ID**: 3

**Level:** High

**Description:** Crystal growth is very fascinating. So I have decided to check how the crystal grows if there are small cells of frozen and liquid cells in the shape of a star. Turns out it takes the shape of the butterfly with the help of following logic which is self-explanatory

**if** (getDesiredNeighborsCount(CACellState.***FROZEN***) >1 && **this**.getCellState() == CACellState.***VAPOUR***) {

**return** CACellState.***FROZEN***;

}

**if** (getDesiredNeighborsCount(CACellState.***FROZEN***) == 1 && **this**.getCellState() == CACellState.***VAPOUR***) {

**return** CACellState.***LIQUID***;

}

**if** (getDesiredNeighborsCount(CACellState.***LIQUID***) > 1 ) {

**return** CACellState.***FROZEN***;

}

**if** (getDesiredNeighborsCount(CACellState.***VAPOUR***) > 3 && **this**.getCellState() == CACellState.***LIQUID***) {

**return** CACellState.***FROZEN***;

}

**Actors:** Frozen Cells -> Indicated in blue, Liquid Cells -> Indicated in green

**Stakeholders & Interests:** All the cells in the region

**Pre-Conditions:**

1. Rule to be selected by the user is Butterfly

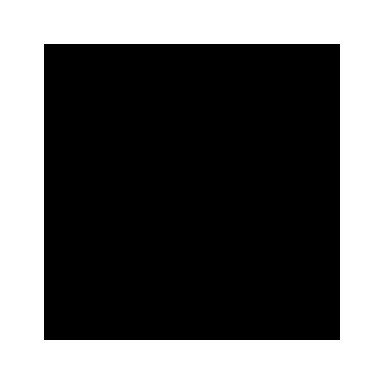
2. Create the initial conditions using Create Button of the side panel.

**Triggers:**

1.  Start Icon -> Start Simulation

2.  Pause Icon -> Pause Simulation

3.  Rewind Icon -> Rewind Simulation

4.  Stop Icon -> Stop Simulation

**Post-Conditions:**

1. **Success**: User is shown “Simulation Completed Successfully” message in status Panel and the console.

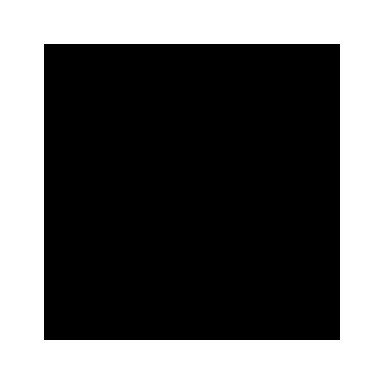
2. **Generation Limit Reached**: When Generation Count reaches Generation Limit, a status “Simulation reached maximum generation Limit…” is shown both in the console and status panel.

3. **Initial State:** When user clicks on  &  and goes back to initial state, Simulation is paused and a status “Simulation Paused as user went back to initial state” is shown. Now User can start or stop the simulation.

**Main Scenario:**

**STEP 1:** Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

**STEP 2:** button is pressed, and simulation is started and finished successfully

**STEP 3:** button is pressed, and simulation is stopped, and simulation buttons are disabled and new set of parameters can be selected

**Alternate Scenarios:**

**Scenario 1:**

STEP 1: Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

STEP 2: button is pressed, and Simulation is started.

STEP 3: button is pressed, and Simulation is paused.

STEP 4:  button is pressed, and Simulation goes back from the generation it was paused to previous generation, reaches initial state and simulation is paused again.

**Scenario 2:**

STEP 1: Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

STEP 2: button is pressed, and Simulation is started.

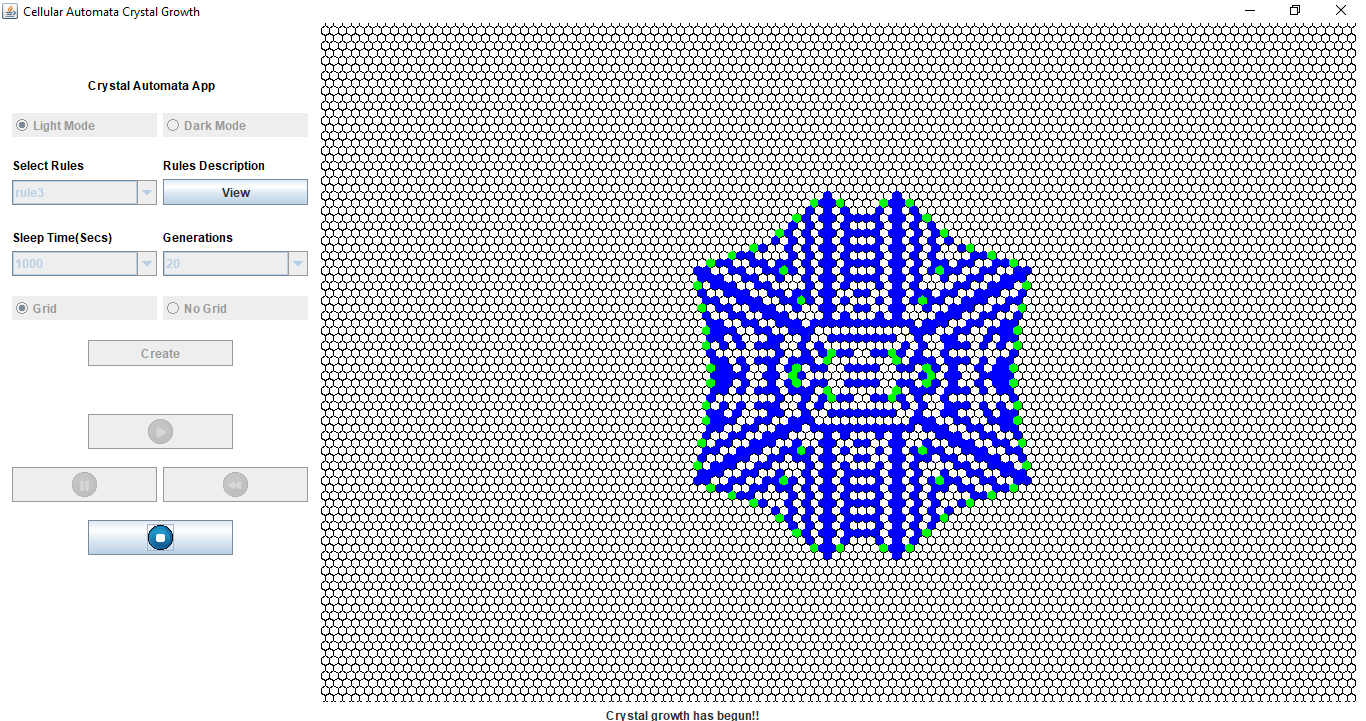
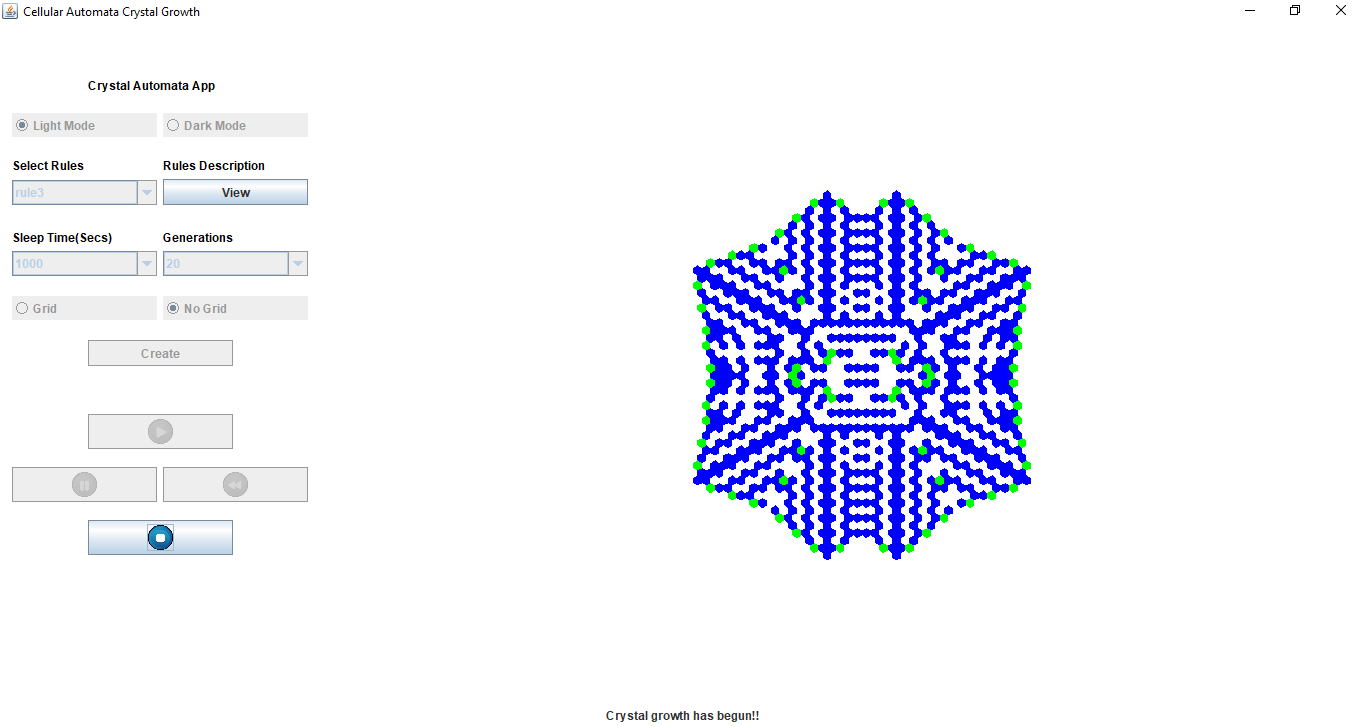
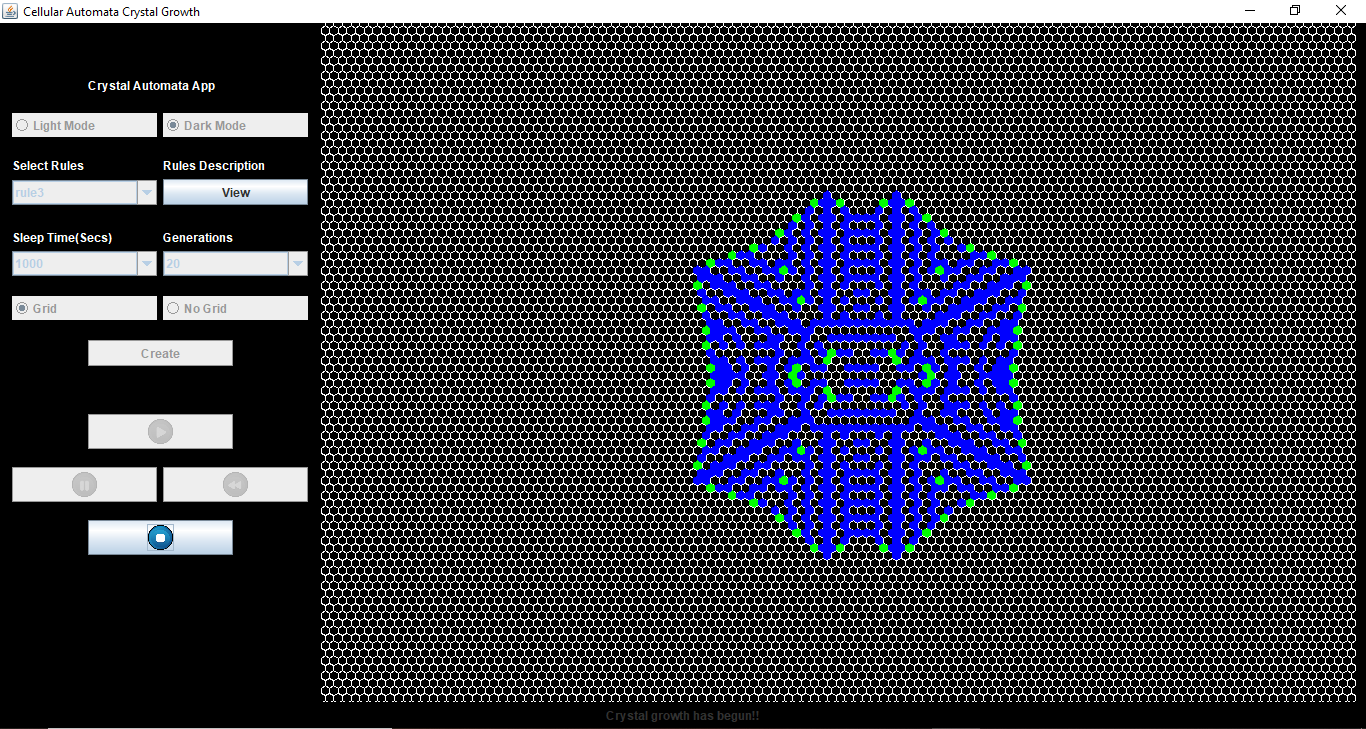
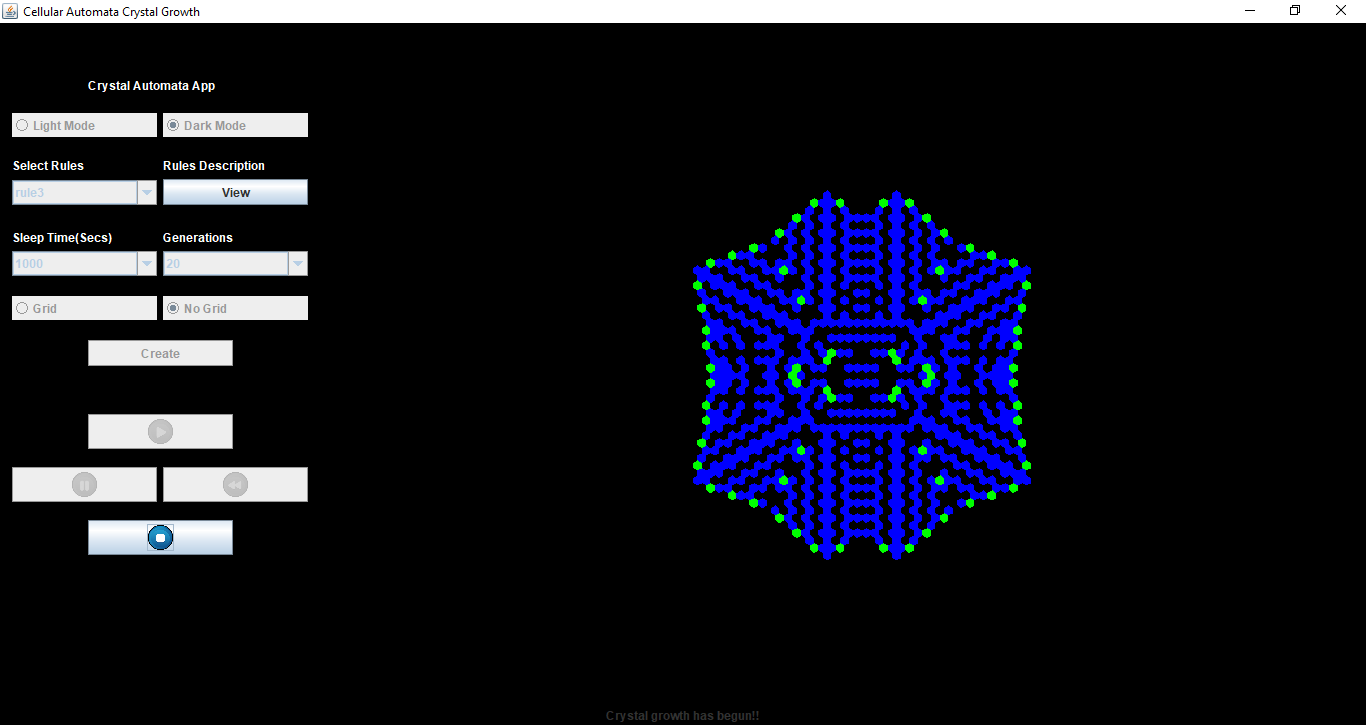
STEP 3: button is pressed, and Simulation is paused.

STEP 4:  button is pressed, and Simulation goes back from the generation it was paused to previous generation.

STEP 5: button is pressed, and Simulation is paused.

STEP 6: button is pressed, and Simulation is started from the point it was paused.

Following are snapshots of the final simulation results with Grid or without grid and in Light/Dark mode



**Use Case 4**: Highway

**ID**: 4

**Level:** High

**Description:** Crystal growth is very fascinating. So I have decided to check how the crystal grows if there are small cells of frozen and liquid cells in the shape of a big cross. Turns out it takes the shape of the highway with the help of following logic which is self-explanatory

**if** (getDesiredNeighborsCount(CACellState.***FROZEN***) >1 && **this**.getCellState() == CACellState.***VAPOUR***) {

**return** CACellState.***FROZEN***;

}

**if** (getDesiredNeighborsCount(CACellState.***FROZEN***) == 1 && **this**.getCellState() == CACellState.***VAPOUR***) {

**return** CACellState.***LIQUID***;

}

**if** (getDesiredNeighborsCount(CACellState.***LIQUID***) > 1 ) {

**return** CACellState.***FROZEN***;

}

**if** (getDesiredNeighborsCount(CACellState.***VAPOUR***) > 3 && **this**.getCellState() == CACellState.***LIQUID***) {

**return** CACellState.***FROZEN***;

}

**Actors:** Frozen Cells -> Indicated in blue, Liquid Cells -> Indicated in green

**Stakeholders & Interests:** All the cells in the region

**Pre-Conditions:**

1. Rule to be selected by the user is Butterfly

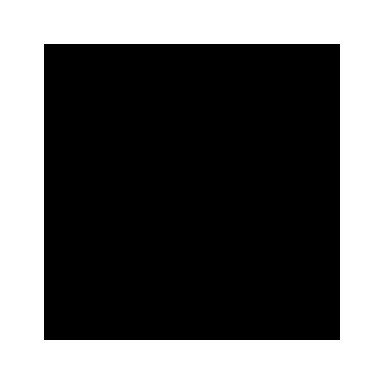
2. Create the initial conditions using Create Button of the side panel.

**Triggers:**

1.  Start Icon -> Start Simulation

2.  Pause Icon -> Pause Simulation

3.  Rewind Icon -> Rewind Simulation

4.  Stop Icon -> Stop Simulation

**Post-Conditions:**

1. **Success**: User is shown “Simulation Completed Successfully” message in status Panel and the console.

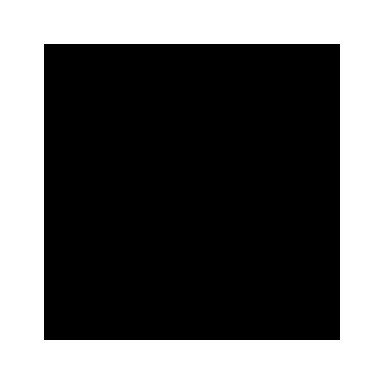
2. **Generation Limit Reached**: When Generation Count reaches Generation Limit, a status “Simulation reached maximum generation Limit…” is shown both in the console and status panel.

3. **Initial State:** When user clicks on  &  and goes back to initial state, Simulation is paused and a status “Simulation Paused as user went back to initial state” is shown. Now User can start or stop the simulation.

**Main Scenario:**

**STEP 1:** Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

**STEP 2:** button is pressed, and simulation is started and finished successfully

**STEP 3:** button is pressed, and simulation is stopped, and simulation buttons are disabled and new set of parameters can be selected

**Alternate Scenarios:**

**Scenario 1:**

STEP 1: Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

STEP 2: button is pressed, and Simulation is started.

STEP 3: button is pressed, and Simulation is paused.

STEP 4:  button is pressed, and Simulation goes back from the generation it was paused to previous generation, reaches initial state and simulation is paused again.

**Scenario 2:**

STEP 1: Select Rule -> Select Light/Dark mode -> Select Sleep Time -> Select Generations -> Grid/No Grid -> Create

STEP 2: button is pressed, and Simulation is started.

STEP 3: button is pressed, and Simulation is paused.

STEP 4:  button is pressed, and Simulation goes back from the generation it was paused to previous generation.

STEP 5: button is pressed, and Simulation is paused.

STEP 6: button is pressed, and Simulation is started from the point it was paused.

Following are snapshots of the final simulation results with Grid or without grid and in Light/Dark mode

